SOFTWARE ENGINEERING & CONCEPTS – LAB MANUAL

E AUTHENTICATION SYSTEM USING OTP GENERATION

SNEKHA R

220701282

B.E COMPUTER SCIENCE AND ENGINEERING
II E

Software Concepts & Engineering - Lab Manual

Overview of the Project

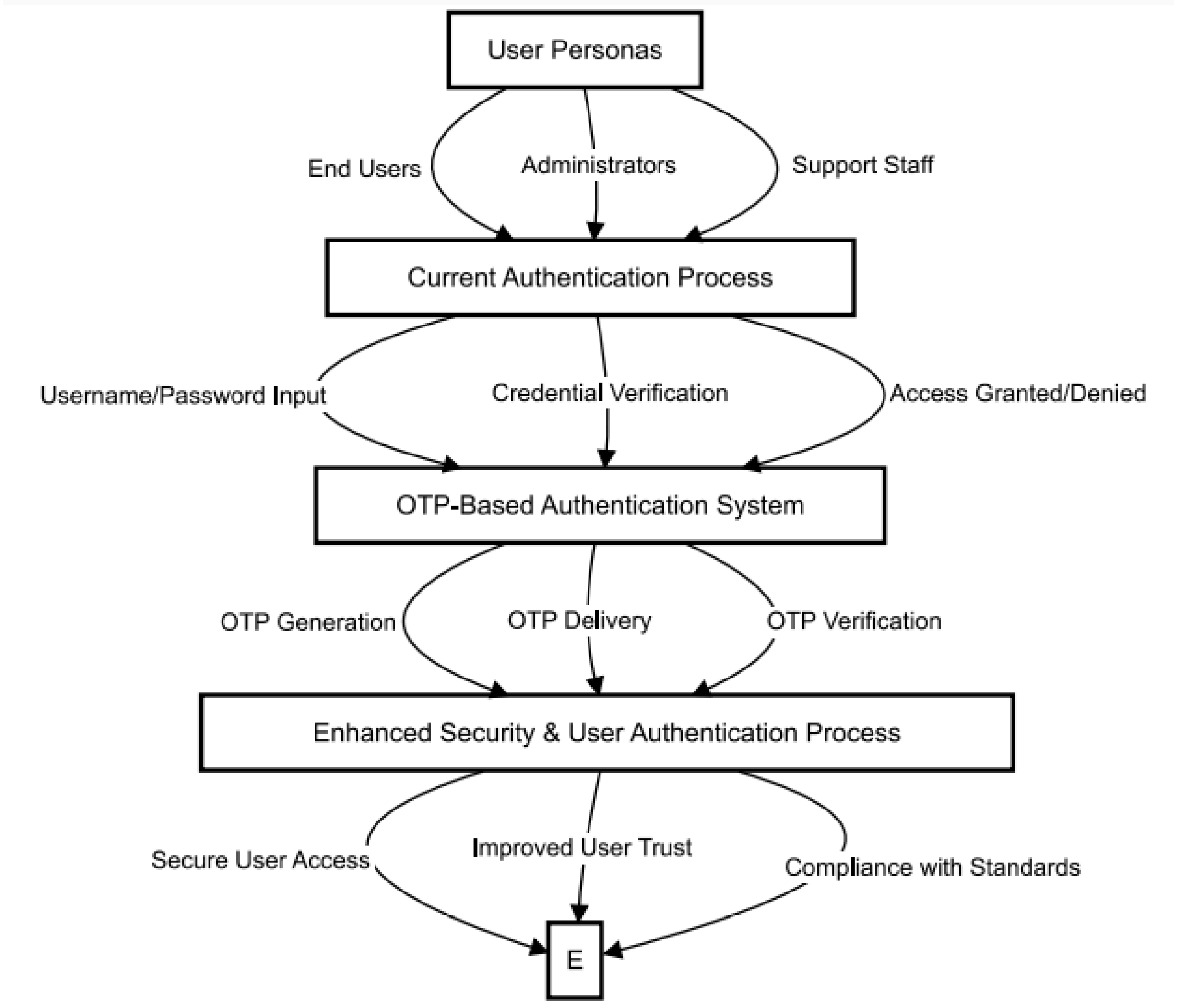
The primary problem this project aims to resolve is ensuring secure and reliable user authentication in online systems. Traditional username and password authentication methods are increasingly vulnerable to security breaches, such as phishing attacks, password theft, and brute force attacks. Users often reuse passwords across multiple sites, exacerbating the risk of unauthorized access.

The e-authentication system with OTP (One-Time Password) generation addresses these vulnerabilities by adding an extra layer of security. OTPs are dynamically generated and valid for a short period, making them more secure against interception and reuse.
Implementing an e-authentication system using OTP generation significantly enhances security for users, reducing the risk of unauthorized access even if their passwords are compromised. OTPs, which are valid for a short period and are not reusable, offer robust protection against phishing attacks aimed at capturing user credentials. This advanced security measure increases user confidence and trust, as users are more likely to trust a system that prioritizes their safety. Knowing their accounts have an additional layer of security provides users with peace of mind. Moreover, this system ensures compliance with various security regulations and standards, such as GDPR, HIPAA, and PCI-DSS, which mandate strong authentication mechanisms.

The process of receiving and entering an OTP is straightforward and user-friendly, making the added security step simple to navigate. Users benefit from the flexibility of receiving OTPs through multiple channels, such as SMS, email, or app notifications, enhancing convenience. Additionally, in the event of a password compromise, the requirement of an OTP for login mitigates the impact, as the attacker would also need access to the user's OTP delivery channel. By implementing this system, organizations can offer a more secure authentication method, substantially reducing the risk of account compromises and strengthening the overall security posture of their online services.

Business Architecture Diagram

The primary business need for implementing an e-authentication system using OTP generation is to enhance the security of user authentication processes. This need arises from the increasing prevalence of cyber threats, such as phishing, credential stuffing, and brute force attacks, which compromise user accounts and sensitive data.



Current process:
1.Manual Authentication:

- User Input: Users enter their username and password.
- Credential Verification: The system verifies these credentials against the stored database.
- Access Decision: Based on the verification, users are either granted or denied access.

2.Automatic Authentication:

- Password Storage: User passwords are stored in a hashed format in the database.
- Login Attempts: When users attempt to log in, the system automatically verifies the entered credentials.

- Access Logs: The system logs all access attempts for monitoring purposes.

Personas and Their Interactions with the System:
1.End Users:
- Current Process: Users log in using their username and password.
- New Process: Users will receive an OTP via SMS, email, or a mobile app after entering their username and password. They will need to enter this OTP to complete the login process.

2. Administrators:
- Current Process: Admins manage user accounts and handle password resets manually or through automated systems.
- New Process: Admins will manage the OTP generation and delivery systems, ensure the integrity of the authentication process, and monitor security logs for suspicious activities.

3. Support Staff:
- Current Process: Support staff assist users with login issues, including password resets and account recovery.
- New Process: Support staff will also assist with issues related to OTP delivery and verification, providing guidance on how to receive and use OTPs.

Business Problems

1. Security Vulnerabilities:
- Problem: The current authentication process relying solely on usernames and passwords is susceptible to various attacks, including phishing and brute force attacks.
- Solution: Implementing OTP adds a second layer of security, significantly reducing the risk of unauthorized access.

2. User Trust and Confidence:
- Problem: Users may lack confidence in the security of their accounts, leading to reduced trust in the service.
- Solution: Enhanced authentication methods, such as OTP, can improve user trust by providing a more secure login process.

3. Regulatory Compliance:
- Problem: Failing to meet security standards and regulations can result in legal consequences and loss of reputation.
- Solution: OTP-based authentication helps comply with security regulations like GDPR, HIPAA, and PCI-DSS, ensuring that the business meets necessary compliance requirements.

4. Operational Efficiency:
- Problem: High volumes of password reset requests can burden support staff and increase operational costs.
- Solution: With OTP-based authentication, the need for frequent password resets may decrease, as users will benefit from an additional layer of security, reducing the likelihood of compromised accounts.

Requirements as User Stories

Epic : User Login
- Feature:Login with Username and password
    1. As a user, I want to log in to the system using my username and password so that I can access my account.
       Acceptance criteria:
        1.The system should provide a login screen with fields to enter username and password.
        2.The system should validate the username and password. Error messages should be displayed for:
          - Empty username or password field.
          - Invalid username (not found in the system).
          - Incorrect password.
        3.The password field should obscure the entered characters (use asterisks or dots).

       Task:Design Login Attempt
       Task:Handle Login Attempt

       Estimate : 8 points

    2. As a user, I want to be able to reset my password if I forget it so that I can regain access to my account
       Acceptance Criteria:
        1. The new password should meet pre-defined complexity requirements (minimum length, character types).
        2. The reset link or temporary password should be time-bound and expire after a set period (e.g., 24 hours) to prevent unauthorized access attempts.

       Task:Design Password Reset Request
       Task:Send Password Reset Email
       Task:Handle Password Reset Link/Token
- Feature:User Profile Management

3. As a user, I want to be able to update my profile information (e.g., email, phone number) so that my contact details are always current and correct.
   Acceptance criteria:
   1. The system should provide a screen for updating profile information.
   2. The system should validate the new information before saving.
   3. Users should receive a confirmation email/SMS after updating their profile information.

   Task:Design Profile Update Interface
   Task:Implement Profile Information Validation

   Estimate:5 points

Epic : OTP generation and delivery

- Feature:OTP generation
  4. As a system, I want to generate a unique and time-bound OTP upon a user's login attempt so that users can securely access the system with an extra layer of authentication.
     Acceptance criteria:
     1.Upon a valid username and password entry during login, the system should trigger the generation of a unique OTP.
     2. The generated OTP should be time-bound, expiring after a pre-defined period (e.g.,1 minute).
     3. The system should display an error message if the entered OTP is incorrect or expired.
     4. The system should only grant access to the user's account after successful verification of the entered OTP along with the correct username and password.
     5. The system should offer the option to resend the OTP if the user doesn't receive it within a reasonable timeframe.

     Task:Generate unique OTP
     Task:Set OTP validity period

- Feature:OTP Delivery
  5. As a user I want to choose the OTP delivery method so that I can receive my one-time password through my preferred channel for secure login.
     Acceptance criteria:
     1. Upon triggering OTP generation after valid username and password entry, the system should deliver the code through the user's chosen channel.
     2. The system should provide options for users to choose their preferred method for receiving OTPs during login.

Task:Design OTP Delivery preference section
Task:Utilize user preference during OTP generation
Task:Display delivery confirmation

6. As a system, I want to deliver the generated OTP to the user's registered email address. (Alternatively, SMS) so that user receives the OTP securely and conveniently through their desired channel
   Acceptance criteria:
   1. The system should send an email containing the OTP to the user's registered email address.
   2. The email subject line should clearly indicate it contains a login OTP and body should display the generated OTP.
   3. The system should send an SMS containing the OTP to the user's registered phone number.
   4. The system should ensure secure delivery of the OTP through chosen channels.
   5. The system should display a message on the login screen informing the user that the OTP has been sent to their chosen channel.

   Task:Deliver OTP based on user preference
   Task:Share OTP for validation
7. As a system, I want to limit the number of OTP resend requests within a certain timeframe so that I can prevent abuse and potential security risks.
   Acceptance criteria:
   1. The system should track the number of resend requests per user.
   2. After reaching the limit (e.g., 3 requests within 10 minutes), the system should disable the resend option temporarily.
   3. The system should inform the user when they can request a new OTP

   Task:Track OTP resend request
   Task:Implement resent request limit
   Estimate:3 points

Epic : OTP validation and security

- Feature:OTP validation
  8. As a system, I want to validate the user-entered OTP against the generated otp so that only authorized users with valid codes can access the system.
     Acceptance criteria:
     1.The system should compare the user-entered OTP with the internally generated OTP.
     2. Upon successful validation (matching codes), the system should grant access to the user's account.
     3. If the entered OTP is incorrect or expired, the system should display an error message and deny access.

4. The system should limit the number of consecutive incorrect OTP attempts

Task:Capture user Entered OTP
Task:Validate OTP Match
Task:Handle Failed Attempts
Task:Invalidate Used OTP

- Feature:Security Management
  9. As an administrator, I want to be able to view a log of login attempts, including successful and failed ones so that I can monitor suspicious activity and identify potential security breaches.
     Acceptance criteria:
     1. The system should display a comprehensive login attempt log.
     2. The log should include details like username, timestamp, IP address, login status (success/fail), and used OTP (if applicable).
     3. The administrator should be able to filter and search the log based on specific criteria.

     Task:Design Login Attempt Log View
     Task:Implement Log Search and filter
     Task:Secure Access control

POKER PLANNING:
1. Login with Username and Password (3-5 points): This is a basic functionality and shouldn't be complex to implement.
2. Reset Password (5-8 points): This requires additional functionalities like sending a password reset email and validating the reset token.
3. Update Profile Information (3-5 points): Similar to login, this involves a standard form update process.
4. Generate Time-bound OTP (5-8 points): This involves logic for OTP generation and potentially integrating with a third-party service for time-bound codes.
5. Choose OTP Delivery Method (3-5 points): This depends on the complexity of the UI for choosing methods (e.g., dropdown vs separate toggles).
6. Deliver OTP via Email/SMS (3-5 points): This assumes integration with an email/SMS service is already available.
7. Limit OTP Resend Requests (3-5 points): This involves implementing logic to track and limit resend attempts.
8. Validate User-entered OTP (3-5 points): This requires checking the entered code against the generated one.

9. Login Attempt Log (8-13 points): This involves designing the log structure, storing data, and potentially building functionalities to view and filter logs.

Non-functional Requirements:

1.Security:

- The system shall encrypt all user credentials (username, password) at rest and in transit. This ensures unauthorized access even if data is intercepted. (e.g., hashing passwords)
- The system shall implement measures to prevent brute-force attacks on login attempts. This could involve limiting login attempts after a certain number of failures or implementing CAPTCHAs.
- The OTP generation and validation process shall be secure to prevent unauthorized code generation or interception. This might involve using strong cryptographic algorithms for OTP generation and secure channels for delivery (e.g., HTTPS for email).

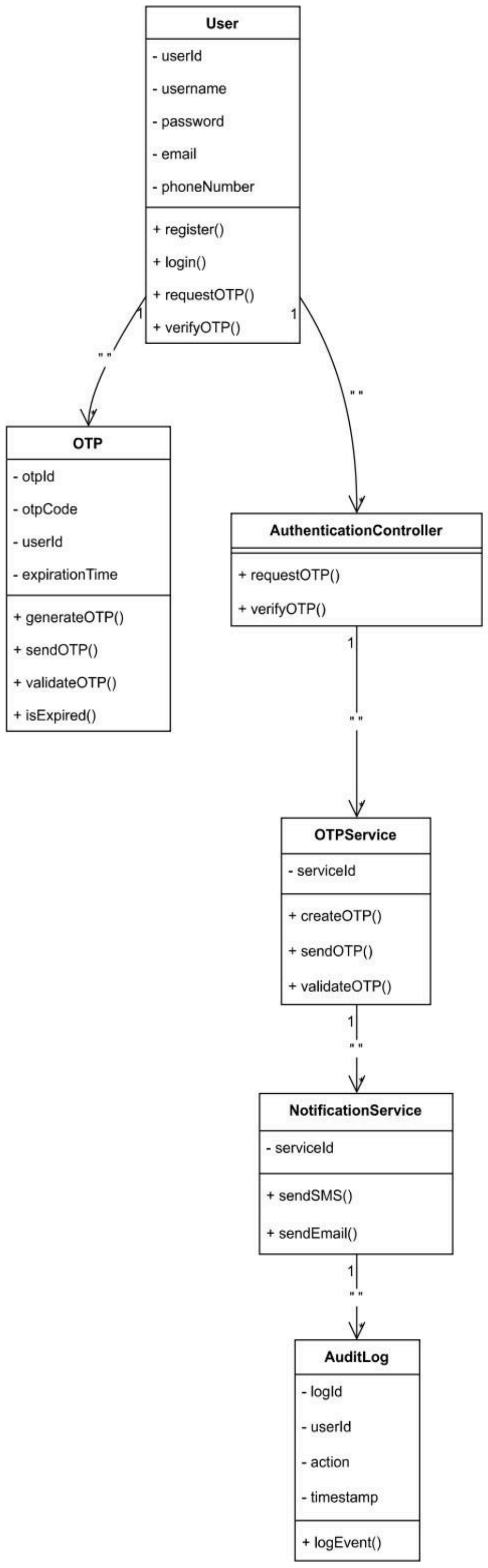
2.Availability:

- The system shall be available for login attempts at least 99% of the time during business hours (excluding scheduled maintenance). This ensures users can access their accounts when needed.
- The system shall have mechanisms for disaster recovery in case of outages. This could involve data backups and redundancy measures.

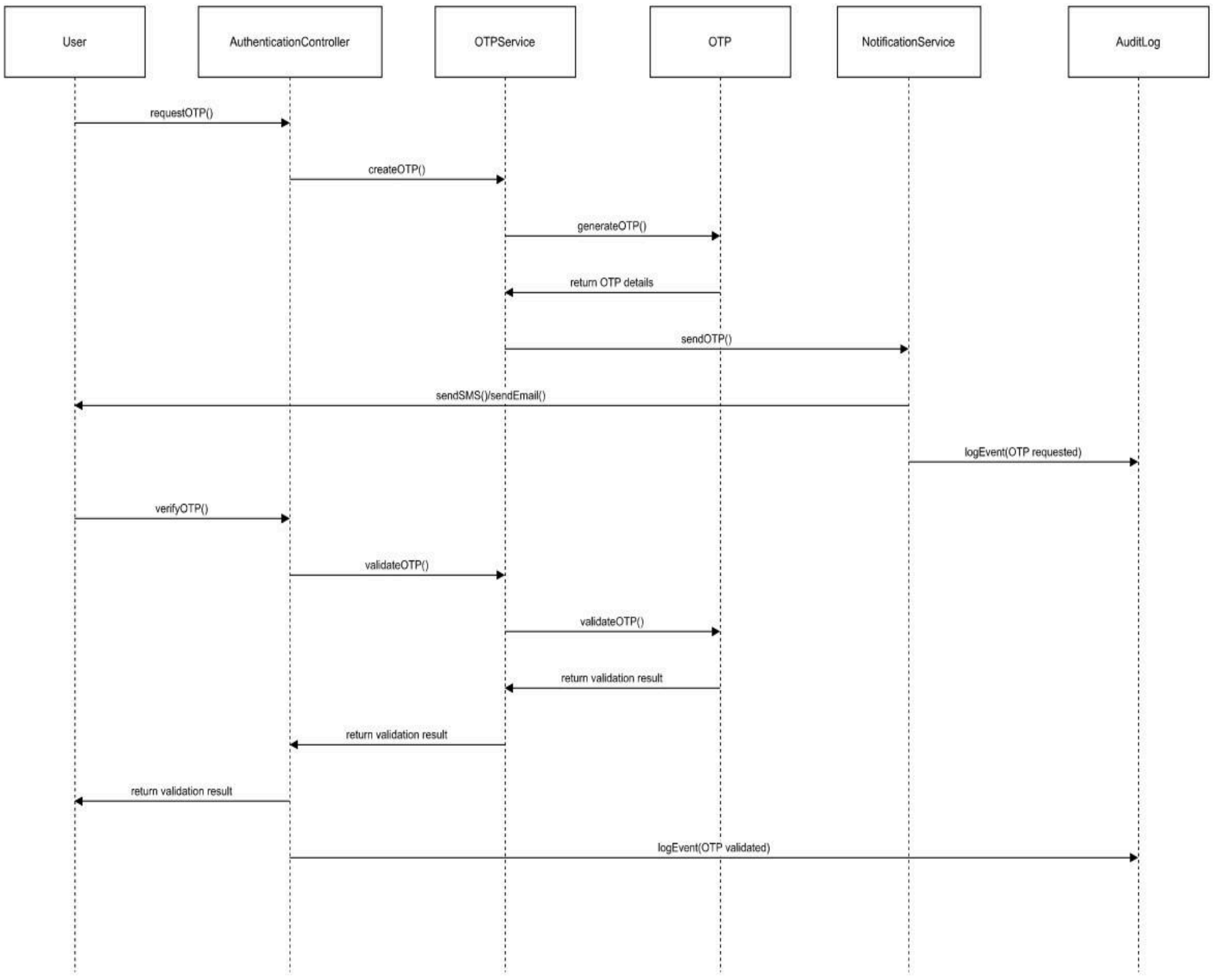
3.Performance:

- The login process, including OTP generation and delivery, shall take no more than 5 seconds on average under normal load. This ensures a smooth user experience.

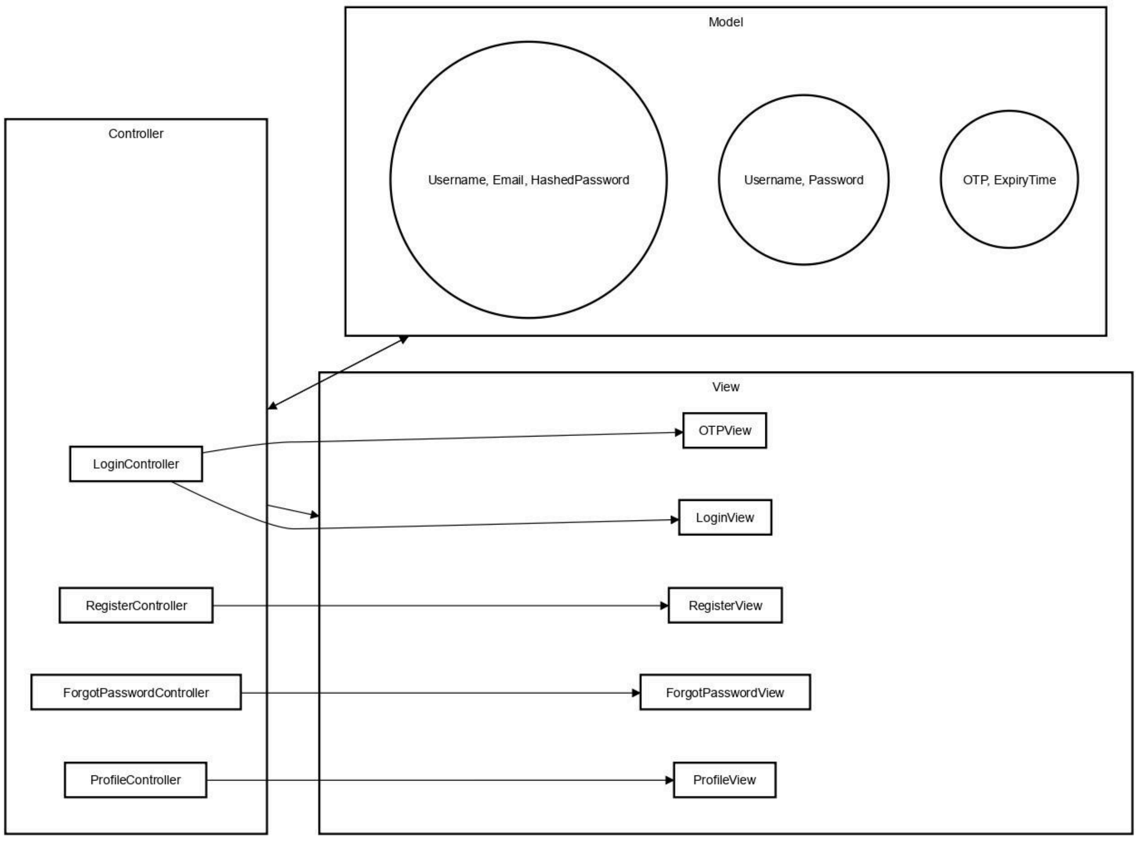
CLASS DIAGRAM:



SEQUENCE DIAGRAM:



Architecture Diagram



Models:

- **User:** Represents a user with properties like username, email, hashed password, etc.
- **LoginModel:** Holds user-provided login credentials (username and password).
- **OTPModel:** Stores the generated OTP and other relevant information (e.g., expiry time).

Controllers:

- **LoginController:** Handles user login requests:
  - Receives login credentials from the login view.
  - Validates credentials against the user database.
  - If valid, generates an OTP and redirects to the OTP view.
  - Handles OTP verification .
  - Upon successful login, creates a session and redirects to the designated application area.
- **RegisterController:** Handles user registration requests (optional).
- **ForgotPasswordController:** Handles "Forgot Password" functionality:
  - Receives the user's email address.

    - Triggers a password reset email sending logic (optional).
- **ProfileController:** Handles user profile management requests (optional).

Views:

- **LoginView:** Provides a form for users to enter their username and password.
- **OTPView:** Displays a field for entering the received OTP.
- **RegisterView:** Presents a registration form for new users.
- **ForgotPasswordView :** Shows a form for entering the user's email to request a password reset.
- **ProfileView :** Allows users to edit their profile information.

Test Strategy

Objective: Ensure the system provides secure and functional user authentication with optional two-factor authentication.
Test suites:
1. **Login Functionality:** This suite would group test cases related to user login, including valid login, invalid credentials, and error handling.
2. **Forgot Password:** This suite would encompass test cases for requesting a password reset, receiving the reset email, and resetting the password.
3. **Profile Management:** This suite would include test cases for editing user profile information, handling invalid data, and database updates.
4. **Two-Factor Authentication :**This suite would focus on testing the OTP generation, delivery, and verification process.
5. **Security:** This suite would encompass test cases for secure password storage, session management, and input validation

Test Suite 1: Login Functionality

Test Case 1.1: Login

- **Steps:**
    1. User enters a valid username and password.
    2. System verifies credentials against the database.
    3. System grants access to the application.
- **Expected Outcome:** User successfully logs in and accesses the application.

Test Case 1.2: Invalid Username

- **Steps:**
    1. User enters an invalid username.
    2. System displays an error message indicating an invalid username.
- **Expected Outcome:** System prevents access and informs the user about the invalid username.

Test Suite 2: Forgot Password

Test Case 2.1: Password Reset

- **Steps:**
  1. User enters a registered email address in the "Forgot Password" form.
  2. System validates the email address.
  3. System generates a password reset token and sends an email containing the reset link.
  4. User receives the password reset email and clicks on the link.
  5. System validates the token and redirects the user to the password reset form.
  6. User enters and confirms a new password.
  7. System updates the user's password and redirects them to the login page with a success message.
- **Expected Outcome:** User successfully resets their password and can log in with the new password.

Test Suite 3: Profile Management

Test Case 3.1: Update Profile Information

- **Steps:**
  1. User logs in to the system.
  2. User navigates to the "Profile" section.
  3. User edits their profile information (e.g., email address).
  4. User submits the changes.
  5. System validates the updated information.
  6. System updates the user's profile information in the database.
  7. System displays a success message to the user.
- **Expected Outcome:** User information is updated successfully, and a confirmation message is displayed.

Test Suite 4: Two-Factor Authentication

Test Case 4.1: OTP Login with email

- **Steps (assuming email for OTP):**
  1. User enters a valid username and password.
  2. System prompts for the OTP sent to the user's registered email.
  3. User receives the OTP and enters it in the designated field.
  4. System validates the OTP and grants access to the application.
- **Expected Outcome:** User successfully logs in after verifying the OTP.

Test Case 4.2 OTP login with SMS delivery

- **Steps (assuming SMS for OTP):**
  1. User enters a valid username and password.
  2. System prompts for the OTP sent to the user's registered phone number.
  3. User receives the OTP and enters it in the designated field.
  4. System validates the OTP and grants access to the application.
- **Expected Outcome:** User successfully logs in after verifying the OTP.

Test Suite 5: OTP Validation

This test suite focuses on the functionality of the OTP (One-Time Password) validation process in your e-authentication system.

Test Case 5.1. Happy Path OTP Validation

- **Description:** This case tests a successful login scenario with valid OTP verification.
- **Steps:**
  - User enters a valid username and password.
  - System prompts for the OTP sent to the user's registered email/phone (depending on configuration).
  - User receives the OTP and enters it in the designated field.
  - **Test Step:** System validates the entered OTP against the stored value.
- **Expected Outcome:**
  - The system verifies the OTP successfully.
  - User gains access to the application.

Test Case 5.2. Invalid OTP

- **Description:** This case tests the system's behavior when an incorrect OTP is entered.
- **Steps:**
  - User enters a valid username and password.
  - System prompts for the OTP.
  - User enters an **invalid** OTP.
  - **Test Step:** System validates the entered OTP against the stored value.
- **Expected Outcome:**
  - The system identifies the entered OTP as invalid.
  - System displays an error message indicating the incorrect OTP.
  - The system may offer options to resend the OTP or retry login after a certain time.

Test case 5.3 Expired OTP

- **Description:** This case tests the system's handling of an expired OTP.
- **Steps:**
  - User enters a valid username and password.
  - System prompts for the OTP.
  - User receives the OTP but **waits beyond the expiry time** before entering it.
  - **Test Step:** System validates the entered OTP against the stored value.
- **Expected Outcome:**
  - The system identifies the entered OTP as expired.
  - System displays an error message indicating the OTP has expired.
  - The system may offer options to resend the OTP or retry login after a certain time.

Deployment Architecture of the application

The deployment architecture for your e-authentication system with OTP generation, leveraging Azure services:

Components:

1. **Web Application:**
   - This is your front-end application that users interact with for login, registration, profile management, etc. It can be built using frameworks like ASP.NET Core, Node.js (Express), or a single-page application (SPA) framework (React, Angular).
   - Deployment options: Azure App Service (for web apps) or Azure Functions (for serverless functions).
2. **API Gateway (Optional):**
   - If your application has multiple backend APIs, an API Gateway can serve as a single entry point, managing routing, security, and other API management functionalities.
   - Deployment option: Azure API Management.
3. **Authentication Service:**
   - This service handles user login, registration, password resets, and session management. It stores user credentials securely (hashed and salted passwords).
   - Deployment option: Azure App Service (for a dedicated backend) or Azure Functions (for serverless functions).
4. **User Database:**
   - This stores user information (username, email, hashed password, etc.).
   - Deployment option: Azure Cosmos DB (NoSQL database for scalability and flexibility) or Azure SQL Database (relational database for structured data).
5. **OTP Generation Service :**
   - This service generates random OTPs and sends them to user-registered email or phone numbers (depending on configuration).
   - Deployment option: Azure Functions (serverless and scalable for handling OTP generation requests).
6. **Email/SMS Delivery Service:**
   - This service can be an external provider like SendGrid, Twilio, or a similar service that integrates with Azure for sending OTPs via email or SMS.
   - Deployment option: External service integrated with Azure Functions (for triggering email/SMS sending).
7. **Session Store:**
   - This stores user session data after successful login. Consider using a distributed cache like Azure Redis Cache for scalability and high availability.
8. **Azure Active Directory (Optional):**
   - If you want to integrate with an existing identity provider, Azure AD can centralize user authentication and authorization across multiple applications.

Deployment Considerations:

- **Resource Groups:** Organize your Azure resources (services, databases, etc.) into logical groups for better management.
- **Security:** Implement secure practices like HTTPS for communication between components, role-based access control (RBAC) for restricting access, and encryption for sensitive data at rest and in transit.
- **Scalability:** Choose services that can scale to meet your application's expected user base and traffic.
- **Monitoring & Logging:** Set up Azure Monitor for application and service health monitoring. Use Azure Application Insights for detailed performance insights and debugging.